Project 1. Image Filtering and Fourier Transform

Submitted By:-Harshdeep Singh Chawla

***Abstract*—Image filtering is a technique by which one can modify or enhance the image. On the other hand, Fourier Transform is a technique by which one can convert a image from spatial domain to frequency domain. Fourier transform is a tool used to decompose an image into its sine and cosine component. The main objective of this project were to become familiar with Matlab and image processing functions,understand image filtering in the space domain and to learn to compute and interpret the 2-D Fourier transform of an image. The questions asked in the project full-fills the objective as we get familiar with Matlab and get use a lot of tools used in image processing.**

***Index Terms*—Image Filtering, Noisy Image, SNR, 2-D DFT, 2-D IFT,Mean Square Error.**

I. Introduction

This project, is about becoming familiar with Matlab and in the domain of Image Processing. The project can be divided into six parts.The first part is about filters which includes use of Matlab's standard filters, designing a filter through our own code,checking the time taken to filter an image and plotting frequency response of various filter used. The second section is about adding noise to an image and after filtering checking the SNR i.e. signal to noise ratio of the two images(original and filtered). The third section is about detection of edges. The other three sections are concerned with 2-D DFT and 2-D IFFT of the images. In fourth section we see how omitting some of the frequency affect an image. We also get to calculate MSE i.e. Mean Square Error of the images. The Last section of the project deals with reconstruction of a two new images by interchanging their magnitudes.

II. Filtering in Space Domain

A. Image Filtering using Matlab commands.

An image is read using imread() command, imshow command is used to display the read image. to create a filter kernel fspecial command was used. The order of the filter kernel was kept five. the filter kernel was applied to the image using filter 2 command.The resulting image was more smooth. Freqz2() command was used to determine the frequency response of the filter kernel. All the images were than displayed using imshow , figure, subplot commands. Plot command was used for plotting frequency response of the filter.

When the filter was applied over the same image again and again using for loop, the image become blurry.

Harshdeep Singh Chawla,Department of Computer Engineering, Rochester Institute of Technology, Rochester, NY 14623, USA (e-mail: hxc3427@g.rit.edu).

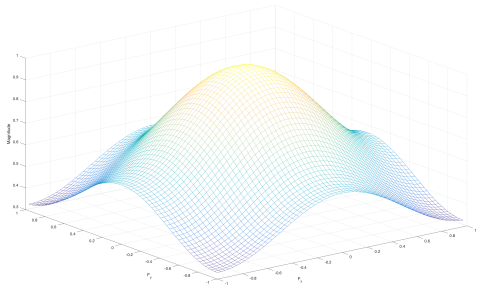

Figure 1. Frequency response of LPF

B. Image Filtering by Writing your own Code.

An image is read using imread() command. Now to design a high pass FIR filter, transfer function of Butterworth filter was taken into order. H (u, v) = 1-(1 /1+[D(u,v)/D]^2n) here, H(u,v) is the high pass filter which was designed as h= 1-1./(1+((((sqrt((u).^2 +(v).^2))./10).^10)); values for u and v were provided using meshgrid command. Freqz2() command was used to see the frequency response of the filter, which was plotted using plot() command. The other images were displayed using imshow() and subplot command.

The final output was dark on the edges and got light on the other part of the original image. This is because most of the low frequencies were omitted y the filter.

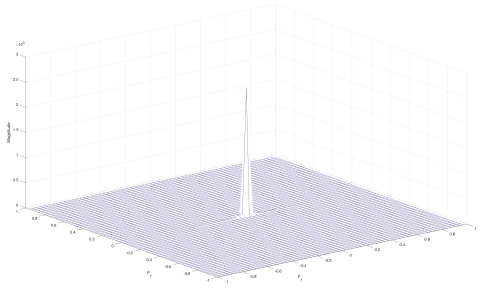

Figure 2. Frequency Response of HPF

C. Timing the Result

In this section, resizing of the original image was done twice. The first new image was half the size of the original image and the second image was twice the size of the original image All the steps done before in section 2a and 2b were repeated for these images. the output was similar as before. But when we apply tic and toc command to all the filtering

Figure 3. Final results of Section 2

functions, it was observed that image shorter image filtering was executed faster as compared to image of bigger size.

Moreover, in the same size images while using two different methods for filtering,it was observed that the filter which was designed by own coding executed faster the compared to the filter already present in Matlab.

The timing for the operation done in section A is time for matlab filter is 9.089442e-03, time for coded filter is 8.193940e-04

Image Size	*Timing with Matlab's standard Filter*	*Timing with High Pass Butterworth Filter*
Original Image(512X512)	1.056920e-02	1.062360e-03
Half Of Original Image(256X256)	7.491112e-03	5.689340e-04
Twice Of Original Image(1024X1024)	1.266972e-02	4.716401e-03

Figure 4. Time response comparision of Filters

III. Filtering a Noisy Image

A. Adding noise to an image and calculating SNR

An image was read using imread() command. Firstly salt and pepper noise with 0.05 density was introduced to the image using imnoise() command. Then we introduced gaussian noise in the original image with 0 mean and 0.01 variance this was also done by imnoise() command.

To calculate SNR value, all the three images (original image, image with salt and pepper noise and the image with gaussian noise) were converted to double using im2double() command and were then indexed so that the file is converted to 1-D from 2-D. Indexing was done to calculate variance of the images. variance was calculated using var() command. And then the SNR values were calculated using the formula mentioned in the question.

The SNR values were snr for salt & pepper noise image is -0.655712dB snr gaussian noise image is -0.403973dB.

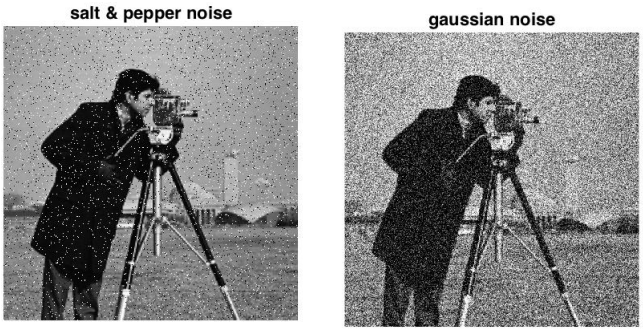

Figure 5. Salt and Pepper Noise; Gaussian Noise

B. Filtering the noisy image

For filtering the noise, we made use of median filter using the command medfilt2() and for low pass gaussian filter[5X5] fspecial() command was used.

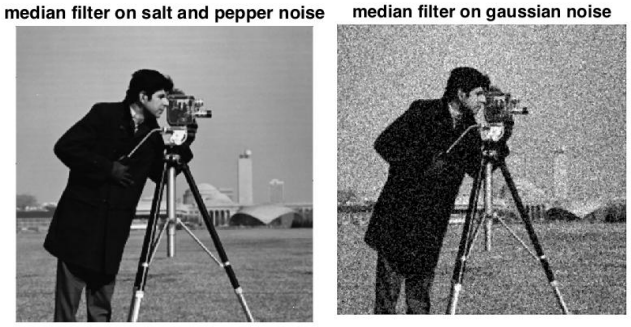

Figure 6. Median Filter on noisy images

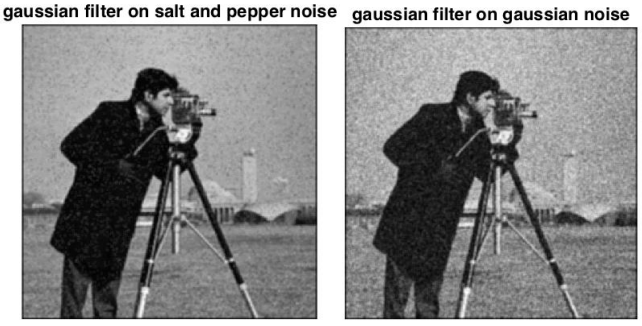

Figure 7. Gaussian filter on noisy Images

The median filter was able to remove the salt and pepper noise but didn't work on gaussian noise.On the other hand the gaussian filter was able to remove the effect of gaussian noise but it didn't work on salt and pepper noise, but it was able to decrease the effect of the noise.

IV. Sobel Edge Detection

the sobel edge detection is used in image processing and computer vision for the purpose of emphasizing edges of an image.

A. Without Using edge command

An image was read using imread(), for the two different edge detections i.e. horizontal and vertical sobel masks were designed. sx=[-1,0,1; -2,0,2; -1,0,1]; sy=[-1,-2,-1; 0,0,0; 1,2,1]; here, sx is used as a horizontal sobel mask and sy is used as a vertical sobel mask. then by using filter2(), these masks were applied on the original image separately. the two results were added and the edges can be seen in the final image.

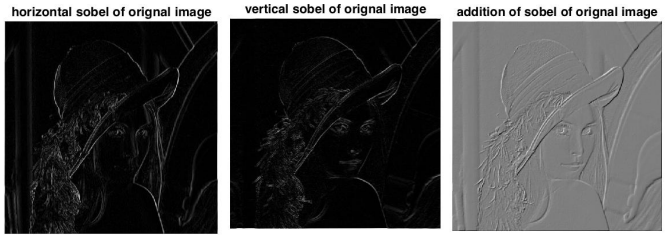

Figure 8. Edge Detection using hrizontal and vertical sobel masks

Then salt and pepper noise and Gaussian noise was added to the original image separately as in the section 3a and the above steps for edge detection were applied again. The output images were not clear for both the cases(gaussian and salt and pepper).

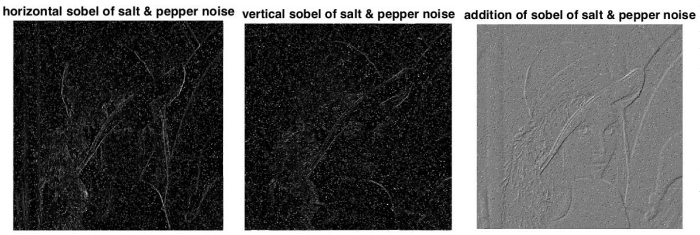

Figure 9. Edge Detection of noisy image(salt & pepper) using hrizontal and vertical sobel masks

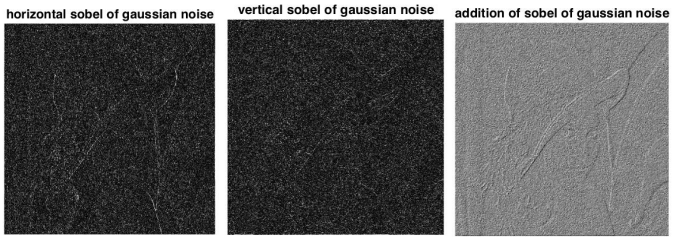

Figure 10. Edge Detection of noisy image(gaussian) using hrizontal and vertical sobel masks

B. Using edge command

For this part, edge() command was used, which directly apply the sobel mask on the image. Since this time no noise was added, the result was much clear and edged can be seen easily.

But when the edge command was used on the noisy image, edges were not displayed properly.

Figure 11. sobel of original image using edge command

C. Comparison between section 4a and 4b

The two edge detection techniques used here, performs same operation on the images. But the output of the mask which was created using the horizontal and vertical components was much better in visibility as compared to the one using edge command. But the edge command requires just one step to perform the execution and the latter requires many steps for the same

D. Method to reduce effect of noise while detecting edges

The noise affects the output of the sobel edge detector very much. The edges are not sharp and get distorted due to noise. Plus the images with the noise were dark but the images without noise were light.

To overcome this problem, median filter can be used on the salt and pepper noise image and gaussian filter can be used over the image with gaussian noise. The output is still dark but the edges are visible.

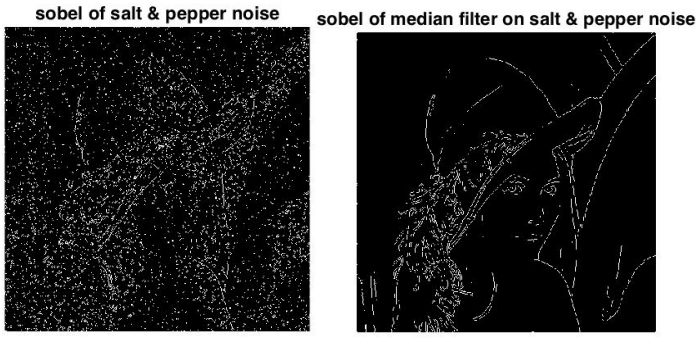

Figure 12. Images before and after using median filter on salt and pepper noise

V. 2-D Fourier Transform of a Sinusoid

In this section , an array(1X512) of zeroes is made using zeroes() command. Next, we generate two sinusoid waves(1X512) ,one is with integral number of cycles and the other without it. The dimensions of the array of zeroes and the sinusoid waves are kept same. Then the two waves are convolved with the transpose of the array one by one. The result of this are two images of 512X512. Then fft2() command is used to take 2D DFT of the images and the

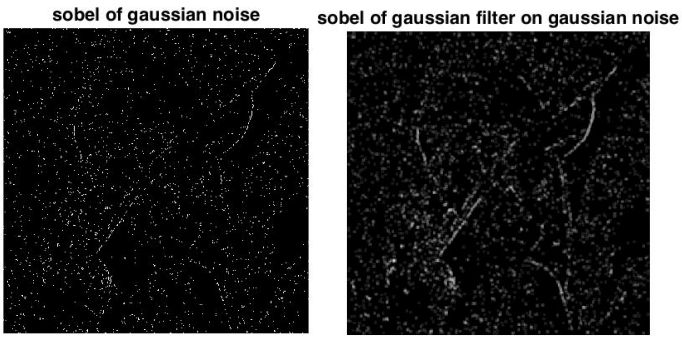

Figure 13. Images before and after using gaussian filter on gaussain noise

magnitudes of their dft is display using abs() command and imshow() command.

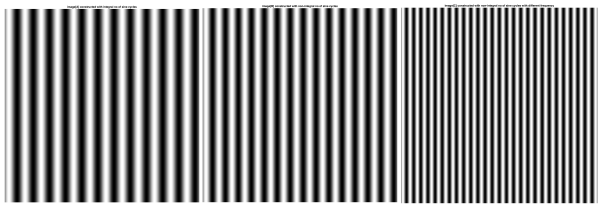

Figure 14. Images Formed By varrying sine wave

By observing the final result, it can be said that, the origin of the 2D DFFT without using shift command occurs at two places. And if fftshift() and log is used the origin shifts to the center and the result is better as the dynamic range is reduced and the spectrum details can be seen.

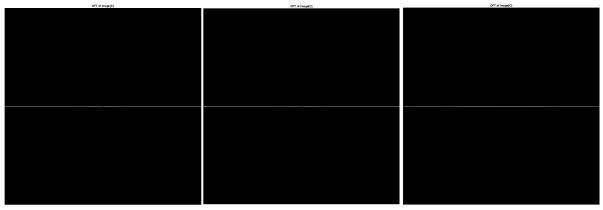

Figure 15. DFT of the Images constructed

VI. The 2-D Fourier Transform of a Real Image

A. Taking 2-D DFT

An image was read using imread() command. The DFT of the image was done using fft2 command which transformed the image from the spatial domain to the frequency domain. The magnitude of the image is displayed using abs() command over the DFT. The origin of the image is located at the center. The magnitude of the image was displayed after using ffshift() command, so that all the zero-frequency component in the middle of the frequency spectrum. The magnitude of the image with log was much better than the one without log because most of the frequency was visible if we take log. So, I would prefer to use log while displaying magnitude.

B. Inverse Fourier Transform and MSE

For taking the inverse Fourier Transform and reconstructing the original image, ifft() command was used over the DFT of the original image. The Mean Square Error, between the original and the reconstructed image, was calculated using the given formula MSE = 1 $\sum x,y\{(f(x,\ y)\ -\ g(x,\ y))2$.. This formula was applied as m=(double(orignal image)-double(ifft2(fft2(original image)))).^2; mse=mean2(m); . mse gives the value of mean squar error which was very less in this case. MSE calculated are MSE of original and ifft of the dft of original image is 2.90142e-28.

Figure 16. Original image and image after DFT

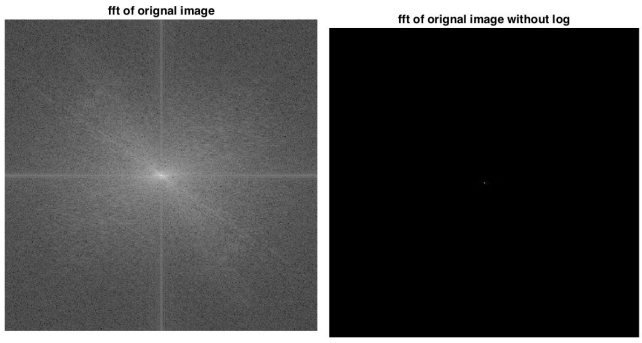

Figure 17. FFT With and Without log

C. Zeroing out Frequency after different radius

For zeroing out all frequencies outside various radius, a circular mask was created. The mask was created using the equation of circle whose radius was varied. The x and y values were set using meshgrid() command. After the filter was designed it was multiplied to the DFT of the original image. And to display the final image inverse Fourier Transform was done of the images.

The difference seen in the output was that as the radius of the circular mask decreased the image became more blurry. This is because most of the frequencies in the image was zeroed.The MSE values were calculated for different masks MSE for N/4 is 35332 , MSE for N/8 is 35269.1 MSE for N/16 is 35155.1 MSE for N/32 is 34941.6.

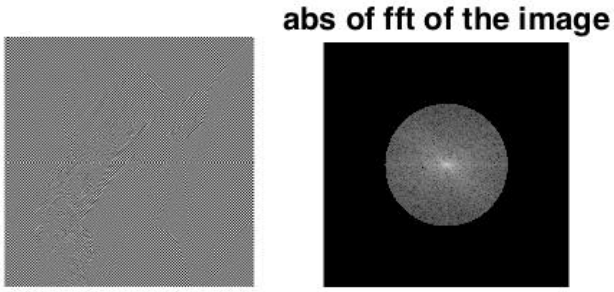

Figure 18. Image for N/4 Mask

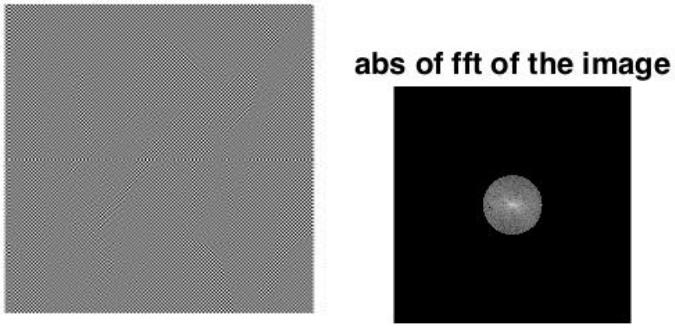

Figure 19. Image for N/8 Mask

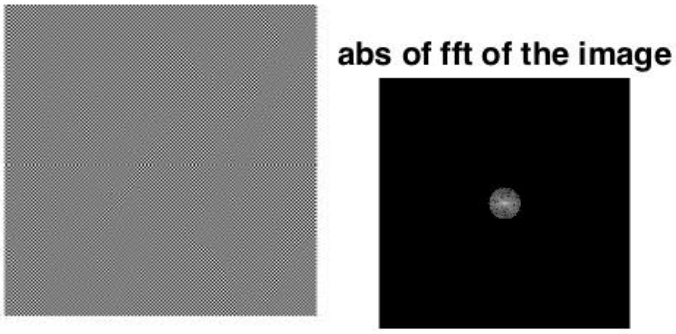

Figure 20. Image for N/16 Mask

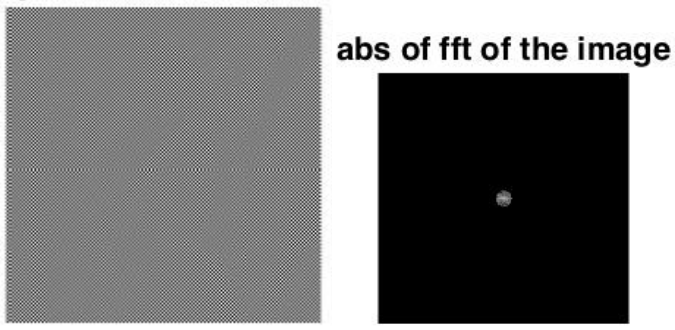

Figure 21. Image for N/32 Mask

VII. The Magnitude and Phase of the 2-D DFT

A. Display the Magnitude and Phase

In this section, two images were read using imshow() command. Then, there Fourier transform were taken using fft2 commond.to shift the origin of the DFT to the center fftshift() was used.As, we know taking DFT of an image will convert the image from spatial domain to frequency domain. Now, to get the magnitude and phase of the image separately, abs() and angle() commands were used.

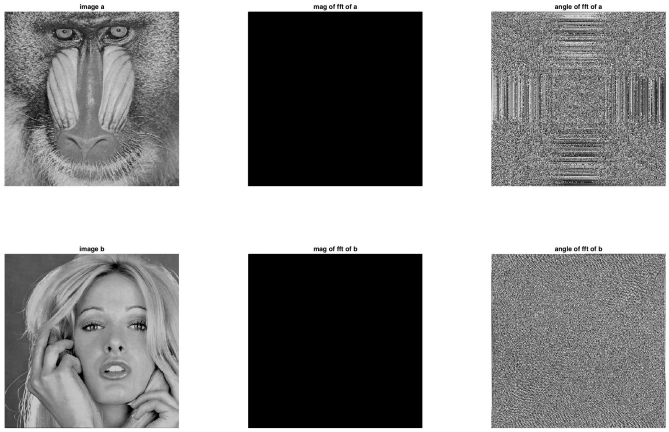

Figure 22. Magnitude and Phase of the Images

B. Reconstruct Image

For the reconstruction of image the magnitude and phase of the images from section 7a were used. The magnitude of the two images were inter changed, and then the magnitude and phase were combined using multiplication or in other words using convolution.

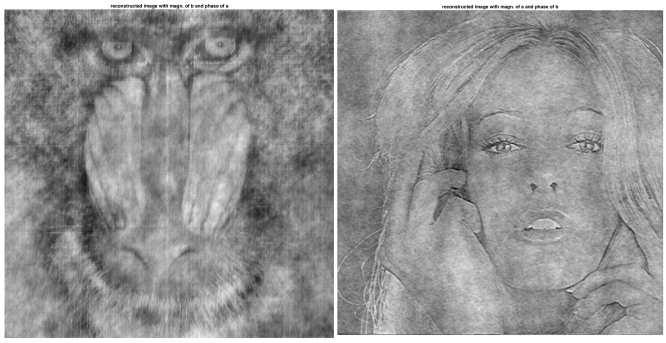

Figure 23. Reconstructed Images

C. Visual Quality of each Reconstruction

As the magnitude of the images reconstructed were inter-changed, the quality of image generated were not that good and as observed, MSE values were very high for both the images. But the images were recognizable and we can say that reconstructing a image using its phase gives good result as compared to using its magnitude. The MSE calculated are MSE for first image is 971.309 MSE for second image is 971.309.

VIII. Conclusion

In this project, we were required to become familiar with Matlab and image processing functions, understand image filtering in the space domain and learn to compute and interpret the 2-D Fourier transform of an image. All the questions asked were answered according to the result obtained. And the the main objective of this image was fulfilled.

Acknowledgment

I would like to thank Professor Andreas Savakis for giving me this project. This project helped me in clearing my doubts and understanding filtering,DFT and image reconstruction in detail. I would also like to thank Professor's TA who helped me during the project.